Kotlin - Interface

In this chapter, we will learn about the interface in Kotlin. In Kotlin, the interface works exactly similar to Java 8, which means they can contain method implementation as well as abstract methods declaration. An interface can be implemented by a class in order to use its defined functionality. We have already introduced an example with an interface in Chapter 6 - section “anonymous inner class”. In this chapter, we will learn more about it. The keyword “interface” is used to define an interface in Kotlin as shown in the following piece of code.

```
interface ExampleInterface {
   var myVar: String     // abstract property
   fun absMethod()       // abstract method
   fun sayHello() = "Hello there" // method with default implementation
}
```

In the above example, we have created one interface named as “ExampleInterface” and inside that we have a couple of abstract properties and methods all together. Look at the function named “sayHello()”, which is an implemented method.

In the following example, we will be implementing the above interface in a class.

```
interface ExampleInterface {
   var myVar: Int        // abstract property
   fun absMethod():String    // abstract method

   fun hello() {
      println("Hello there, Welcome to TutorialsPoint.Com!")
   }
}
class InterfaceImp : ExampleInterface {
   override var myVar: Int = 25
   override fun absMethod() = "Happy Learning "
}
fun main(args: Array<String>) {
   val obj = InterfaceImp()
   println("My Variable Value is = ${obj.myVar}")
   print("Calling hello(): ")
   obj.hello()
```

```
    print("Message from the Website-- ")
    println(obj.absMethod())
}
```

The above piece of code will yield the following output in the browser.

```
My Variable Value is = 25
Calling hello(): Hello there, Welcome to TutorialsPoint.Com!
Message from the Website-- Happy Learning
```

As mentioned earlier, Kotlin doesn't support multiple inheritances, however, the same thing can be achieved by implementing more than two interfaces at a time.

In the following example, we will create two interfaces and later we will implement both the interfaces into a class.

```
interface A {
    fun printMe() {
        println(" method of interface A")
    }
}
interface B  {
    fun printMeToo() {
        println("I am another Method from interface B")
    }
}

// implements two interfaces A and B
class multipleInterfaceExample: A, B

fun main(args: Array<String>) {
    val obj = multipleInterfaceExample()
    obj.printMe()
    obj.printMeToo()
}
```

In the above example, we have created two sample interfaces A, B and in the class named "multipleInterfaceExample" we have implemented two interfaces declared earlier. The above piece of code will yield the following output in the browser.

```
method of interface A
I am another Method from interface B
```